Tribhuvan University

Faculty of Humanities and Social Sciences

"BITE-RUSH"

A PROJECT REPORT

Submitted to:

Department of Computer Application

Janjyoti Multiple Campus

Mahendranagar, Kanchanpur

In partial fulfillment of the requirements for the Bachelors in Computer Application

Submitted by:

Kamal Acharya

Roshan Prasad Acharya

May, 2025

Under the Supervision of

Umesh Raj Pant

Tribhuvan University

Faculty of Humanities and Social Sciences

Janjyoti Multiple Campus

Supervisor's Recommendation

We hereby recommend that this project prepared under my supervision by **ROSHAN PRASAD ACHARYA & KAMAL ACHARYA** entitled "**BITE-RUSH**" in partial fulfillment of the requirements for the degree of Bachelor of Computer Application is recommended for the final evaluation.

…………………

Umesh Raj Pant

Supervisor

Department of Humanities and Social Sciences

Janjyoti Multiple Campus

Mahendranagar, Kanchanpur

Tribhuvan University

Faculty of Humanities and Social Sciences

Janjyoti Multiple Campus

LETTER OF APPROVAL

This is to certify that this project prepared by **ROSHAN PRASAD ACHARYA & KAMAL ACHARYA** entitled "**BITE-RUSH**" in partial fulfillment of the requirements for the degree of Bachelor in Computer Application has been evaluated. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

………………………………… **Umesh Raj Pant** **Supervisor** Janjyoti Multiple Campus Mahendranagar, Kanchanpur	…………………………….. **Puneet Ayer** **Coordinator** Janjyoti Multiple Campus Mahendranagar, Kanchanpur
………………………………… Internal Examiner	……………………………………… External Examiner

ABSTRACT

BITE-RUSH is a food delivery and restaurant management system designed to streamline the process of ordering food online while providing restaurant owners with tools to manage their operations efficiently.Built using modern web technologies (React, Spring Boot), BITE-RUSH ensures a seamless experience for both customers and restaurant owners. The application integrates secure payment gateway (eSewa) and maintains a structured database to handle orders, user accounts, and restaurant details efficiently. The system consists of two main modules: Customer Module in which the customer can browse restaurants and food items,Add items to cart and place orders, Secure online payments via eSewa and Track order status and history; Restaurant Admin Module in which owner can Register and manage restaurant profiles, Create and update food items with customizable ingredients, Process and manage customer orders in real-time and Monitor sales and order fulfillment. This project enhanced my skills in full-stack development, database management, and payment gateway integration, while addressing real-world challenges in the food delivery ecosystem. BITE-RUSH not only simplifies food ordering for customers but also empowers restaurant owners with a robust management system, bridging the gap between demand and supply in the food industry.

ACKNOWLEDGEMENT

Success can't be achieved single handed. It's all about hard work, teamwork and others help. So, I would like to acknowledge all those helping hands in making of this project successful.

Firstly, I would like to express our deep sense of gratitude towards our project guide Umesh Raj Pant sir for your valuable guidance, co-operation and approval towards successful completion of this project work. We are very thankful for their continuous support and constructive suggestion that have enabled this research project to achieve its present form. We are grateful to our H.O.D Dilli Raj Joshi who always solved our problem. We would also like to thank Janjyoti Multiple Campus for providing the resource for research work and providing library and computing support without which knowledge and assistance of this would have not been successful. Last but not least, a great deal of appreciation and best wishes to our teammates as well as to all of my friends for their contribution and encouragement during this work.

We would also like to thank our parents, family, relative, friends for their moral support and encouragement during this project work. However, we accept the sole responsibility for any errors and conflicts that might have occurred in this report.

Project By (Teammates):

Kamal Acharya

Roshan Prasad Acharya

May, 2025

Table Of Contents

ABSTRACT iii

ACKNOWLEDGEMENT iv

List of Abbreviation vii

List Of Figures viii

List Of Tables ix

CHATPER I: INTRODUCTION 1

1.1 Introduction 1

1.2 Problem Statement 1

1.3 Objectives of System 2

1.4 Scope and Limitation 2

1.5 Development Methodology 3

CHAPTER II: BACKGROUND STUDY & LITERATURE REVIEW 5

2.1 BACKGROUND STUDY 5

2.2 LITERATURE REVIEW 5

CHAPTER III: SYSTEM ANALYSIS & DESIGN 7

3.1 System Analysis 7

3.1.1 Requirement Analysis 7

3.1.2 Feasibility Study 9

3.1.3 Data Modeling (E-R Diagram) 9

3.1.4 Process Modelling: DFD 10

3.2 System Design 11

3.2.1 Architectural Design 12

3.2.2 Database Schema Design 12

3.2.3 Interface Design 13

3.2.4: Physical DFD 14

CHAPTER IV: IMPLEMENTATION & TESTING 15

4.1 Implementation 15

4.1.1 Tools Used 15

4.1.2: Implementation Details of Module 15

4.2. Testing 17

4.2.1. Test Case for Unit Testing 17

4.2.2 Test Case for System Testing 18

CHAPTER V: CONCLUSION & FUTURE RECOMMENDATION 19
5.1. Conclusion **19**
5.2. Lesson Learnt/Outcome **19**
5.3. Future Recommendation **19**
References 21
Appendices 22
BIBLIOGRAPHY 31

List of Abbreviation

BCA:	**Bachelor of Computer Application**
DBMS:	**Database Management System**
GB:	**Gigabyte**
JDBC:	**Java Database Connectivity**
ORM:	**Object Relational Mapping**
PC:	**Personal Computer**
RAM:	**Random Access Memory**
SQL :	**Structured Query Language**
SSD:	**Solid State Drives**
WWW:	**Word Wide Web**

List Of Figures

Figure 1.5: Agile Development Model Diagram

Figure 3.1: Use Case Diagram

Figure 3.2: ER Diagram

Figure 3.3: Zero Level DFD

Figure 3.4: Level 1 DFD

Figure 3.5: Architectural Design

Figure 3.6: Database Tables Schema

Figure 3.7: Registration Page

Figure 3.8: Login Page

Figure 3.9: Physical DFD

Figure 3.11: Student Module Screen

List Of Tables

Table 4.1: **Implementation Table**

Table 4.2: **Login/Signup Test Cases**

Table 4.3: **Restaurant and User Review Test Cases**

Table 4.4: **Logout Test Case**

Table 4.5: **System Testing Test Case**

CHATPER I: INTRODUCTION

1.1 Introduction

In the digital age, online food ordering and delivery services have become an essential part of life, providing convenience and efficiency for both consumers and restaurant owners. With the growing demand for food delivery services in Nepal, **BITE-RUSH** aims to develop an Online Restaurant and Food Delivery System that allows users to browse restaurants, explore menus, place orders, and make secure online payments through eSewa, a popular digital wallet in Nepal.

The platform will serve as a bridge between customers and local restaurants, offering a seamless and user-friendly interface for placing food orders. The integration of eSewa ensures safe and real-time transactions, eliminating the need for cash on delivery and making the process more reliable and efficient.

This system is designed to enhance the overall food ordering experience by allowing restaurants to manage orders, menus, and customer feedback, while also enabling delivery personnel to track and fulfill orders efficiently. The project will include web-based functionality for administrators, restaurants, delivery staff, and customers, ensuring a complete and scalable solution for the food service ecosystem.

1.2 Problem Statement

In many places, the process of ordering food from local restaurants is still largely manual, involving phone calls, unclear menu availability, and inconsistent delivery services. This traditional approach often results in poor customer satisfaction due to delays, miscommunication, limited payment options, and lack of transparency in order tracking. Moreover, the absence of a unified digital plaotform for restaurants, customers, and delivery personnel leads to inefficiencies in order management and fulfill. While some global food delivery platforms exist, they are either not available in all regions of Nepal or do not support local payment gateways like eSewa, which is widely used by Nepalese consumers. There is a strong need for a localized, user-friendly, and reliable Online Restaurant and Food Delivery System that supports Nepali digital payment solutions and provides a seamless experience for all stakeholders. This project seeks to address these challenges by creating a complete system that simplifies food ordering, enhances service delivery, and supports secure online payments through eSewa. [1]

1.3 Objectives of System

The main aim of this project is to develop a complete Online Restaurant and Food Delivery System with integrated eSewa payment gateway to provide a smooth and reliable experience for users, restaurant owners, and delivery personnel. The specific objectives include:

- To design & develop a user-friendly web application.
- To allow restaurants to manage their profiles.
- To integrate eSewa for payment.
- To implement admin daashboard to monitor activities.
- To promote digital transformation in local food.

1.4 Scope and Limitation

Scope

The project focuses on developing **BITE-RUSH**, an online food ordering and delivery system for Nepal, with the following boundaries:

- **Customers**: Browse restaurants, place orders, track deliveries, pay online.
- **Restaurants**: Manage menus, update order statuses, and view feedback.
- **Delivery Personnel**: Accept/reject orders and update delivery progress.
- **Admin Dashboard**: Monitor transactions, users, and resolve disputes
- **Web-based** (no mobile app in this phase).
- **eSewa integration** (excludes cash-on-delivery or other payment gateways).
- **Urban focus** (optimized for areas with reliable internet/eSewa adoption).

Limitations

- **Geographical**: Limited to Nepal (may not suit other markets).
- **Payment Dependency**: Only supports eSewa (no credit cards/Khalti).
- **Data Reliability**: Menu/pricing accuracy depends on restaurant updates.
- **Scalability**: High user traffic may require backend upgrades.
- **Delivery Challenges**: No live route optimization (depends on delivery personnel).

1.5 Development Methodology

Agile is an iterative project management and software development approach that helps teams deliver value to their customers faster and with less trouble. [2]

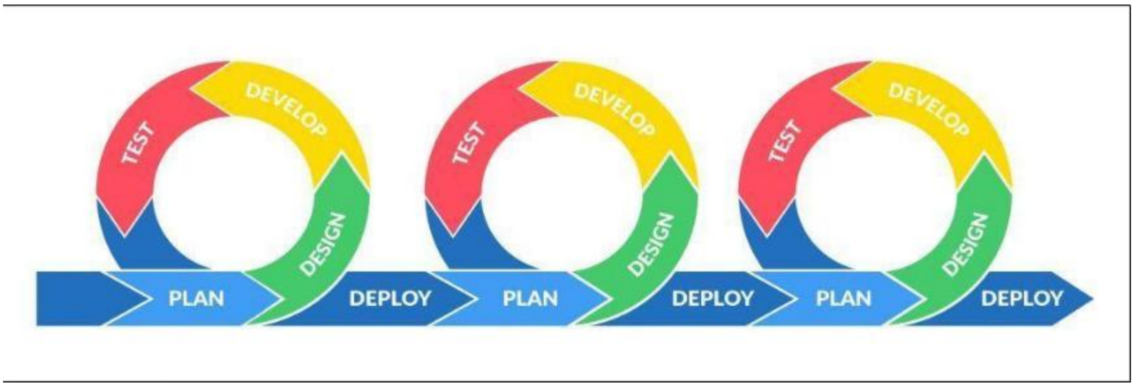


Figure 1.5: Agile Development Model Diagram

1. Requirement Analysis: All the requirements for making Online Food Delivery System in SpringBoot and React including time and resource needs, are listed. These consist of an IDE for coding, a database for storing and retrieving data, and an OS for running the program.

2. Design: The design of this Online Food Delivery System was done using Lucid Chart which is an online tool to design high and low-fidelity images/ diagrams.

3. Development: The development phase consists of coding a function or a module which consists of preparing the backend controllers, services and repositories and designing the frontend user interface. In this, application was broken into several iterations of development and once the development of one module it was tested quickly and then moved to the next iterations.

4. Testing: After development of each module it was tested using Junit, HttpUnit and Mockito and if any test failed it was revised and the problem was found out and tested again until it is solved. It includes different testing phases like dev testing, qa testing and uat testing and finally move to the deployement phase.

5. Deployment: The system was tested and reviewed and verified by the subject teacher i.e., our supervisor through every iteration in the development process. The project was deployed in the system having sufficient hardware and software capabilities

1.6 Report Organization

This project report contains five chapters including this chapter which will cover the designing and development of Bite-Rush.

Chapter I includes the system and the problems, given an overview about the study. It includes introduction, problem of statement, objectives, and solution of the website.

Chapter II covers the literature review which is the previous related work that has been done before. It also includes the description of the website from where we have taken the reference.

Chapter III presents study of system analysis and design of Bite-Rush. In this chapter we can discuss about the project's functional and nonfunctional requirements and also describes the feasibility of the system. And we can design, the system using ER-Diagram and Data flow diagrams. It is main chapter of project development. Here we can design database schema and user interfaces and coding.

Chapter IV discusses the implementation and testing. It includes overall description of the modules that we have implemented into our application. We have shown the test cases.

Chapter V gives the conclusion and future recommendation for the project. This project will be helping justify the importance user requirements. In chapter five, we presented Conclusion and future recommendation respectively. **[3]**

CHAPTER II: BACKGROUND STUDY & LITERATURE REVIEW

2.1 BACKGROUND STUDY

This project implements the idea that a food delivery platform can streamline daily operations for restaurants while enhancing customer convenience through digital ordering, payment integration, and real-time tracking. In Nepal, despite the growing demand for online food delivery, most local restaurants still rely on manual processes (phone calls, walk-ins, or cash payments), leading to inefficiencies such as: order mismanagement due to human errors, Limited payment options, with cash-on-delivery as the default, lack of transparency in order status and delivery timelines. While platforms like Foodmandu and Bhojdeals exist, they often exclude smaller restaurants or lack integration with Nepal's preferred digital wallets (e.g., eSewa). BITE-RUSH addresses these gaps by providing a localized, user-friendly system that benefits all stakeholders. [4]

Application Working Process

Restaurant Admin Module

1. Register/login and create restaurant profile (name, location, cuisine)
2. Add/update/remove food items (prices, descriptions, images)
3. Manage incoming orders (accept/reject, mark as prepared)
4. View order history and customer ratings

Customer Module

1. Browse restaurants/menus with filters (cuisine, price, ratings)
2. Place orders with special instructions
3. Pay via eSewa (no cash handling)
4. Track order status ("Pending" → "Out For Delivery" → "Delivered")
5. Rate restaurants and save favorite orders and restaurants

2.2 LITERATURE REVIEW

We analyzed existing food delivery platforms in Nepal and globally to identify gaps and gather insights for **BITE-RUSH**

To develop **BITE-RUSH**, we conducted an in-depth review of existing food delivery platforms in Nepal and globally to identify industry standards, limitations, and opportunities for innovation. In Nepal, **Foodmandu** dominates the market but faces criticism for its high commission fees (20-30%), which marginalize small local restaurants. Additionally, it lacks integration with Nepal's preferred digital wallets like **eSewa**, forcing reliance on cash payments. **Bhojdeals**, another local platform, offers deals but lacks real-time order tracking and advanced restaurant management tools.

We also examined restaurant management systems like **Dineout Nepal**, which focuses on table reservations rather than delivery, and **Veda POS**, a billing/inventory system that doesn't support online ordering. These limitations highlight the need for a dedicated platform where restaurants can independently manage menus, orders, and customer interactions without third-party restrictions.

For global insights, we analyzed **Zomato (India)** and **Uber Eats**, which offer sophisticated features like live order tracking, dynamic pricing, and multi-payment support. However, their business models are not tailored to Nepal's market, where digital wallet adoption (eSewa, Khalti) outpaces credit card usage. **Uber Eats' real-time GPS tracking** inspired our simplified order status updates, though we adapted it to Nepal's internet reliability constraints.

Payment integration was a key focus—**eSewa's API documentation** was reviewed to ensure seamless, secure transactions, addressing Nepal's shift toward cashless payments. Our research confirmed that no existing platform combines **restaurant self-management, eSewa support, and customer-friendly tracking** in one system. **BITE-RUSH** fills this gap by offering a localized, affordable, and transparent solution for Nepal's food delivery ecosystem. [5]

CHAPTER III: SYSTEM ANALYSIS & DESIGN

3.1 System Analysis

It is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components. This chapter will provide the detail analysis of the current system of Bite-Rush and problems of the current system.

3.1.1 Requirement Analysis

Requirement analysis is significant and essential activity after elicitation. We analyze and refine gathered requirements to make consistent and unambiguous requirements. This activity reviews all requirements and may provide a graphical view of the entire system.

i. Functional Requirement

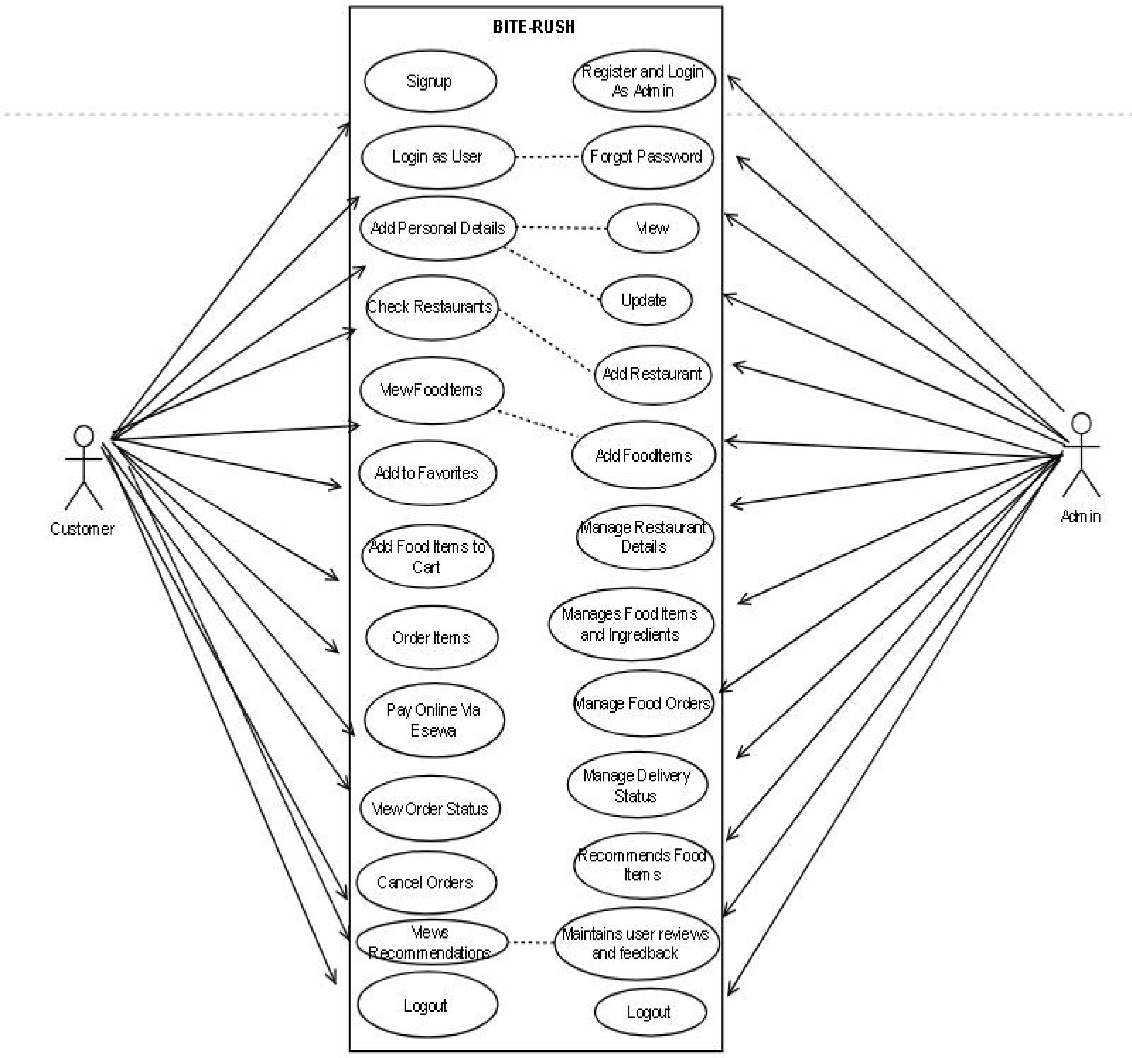


Figure 3.1: Use Case Diagram [6]

The diagram shows two primary actors: **Customer** and **Admin**, representing the system's dual-module structure. Customers begin by signing up or logging in, after which they can browse restaurants, view food items, and add favorites to their profile. The ordering process allows them to add items to their cart, place orders, and make secure payments via eSewa. Customers can track order status, cancel orders if needed, and receive personalized recommendations.

For administrators, the system provides robust management capabilities. After logging in (with password recovery options), admins can add and manage restaurants, update food items and ingredients, and oversee all orders. They update delivery statuses in real-time, recommend popular items, and moderate user reviews. Both actors conclude sessions by logging out, ensuring system security. [7]

ii. Non-Functional Requirement

1. Security: Security is necessary to provide integrity, authentication and availability to the system. In the system, the data taken from the user is safely stored in the PostgreSQL database and can be accessed at any time but only with proper authentication by the authorized person.

2. Reliability: Reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment. The system has been developed using the agile methodology so after each step the system's modules were tested individually as well as at the end so that there is very less possible failure hence the system is considered as reliable.

3. Performance: Performance is an indicator of how well a software system or component meets its requirements for timeliness. The system is pretty fast as the user can easily add their details and in an instance, they can view it as well. Along with it the admin can view the user ordered foods and their feedbacks immediately.

4. Maintainability: The ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment is known as maintainability. The system is flexible and features can be easily added and removed so that it is easy to maintain.

3.1.2 Feasibility Study

In feasibility study, the Bite-Rush Online Food Delivery System's technical, operational, financial & legal factors are analyzed.

i. Technical Feasibility

The main technologies used in Online Food Delivery System as web application are Java, SpringBoot, React JS and PostgreSQL. These technologies are freely available. The technical skills required for the system are sufficient with the development team. The application will not require a huge amount of resources and high-end servers. The system with Intel-I5 8th Gen+ processor, 8GB DDR4 Ram and 256GB SSD is sufficient to execute the project efficiently. It will not require sophisticated technical manpower for operation, as we will initially set up all the run-time environments. Systematic knowledge transfer training accompanied by a user manual will be conducted by us. [8]

ii. Operational Feasibility

The operational implementation is simple and efficient. Once the system is made, the technical skills are not required to manage the web application on a daily basis. Simple knowledge about the system mentioned in the user manual and some simple guidance will be enough for the owner/admin to utilize the features and operate the system i.e, the user of the system can cope up with the system easily without being professional or trained in technical field.

iii. Economic Feasibility

Bite-Rush Online Food Delivery System is a web application, so it requires a server for hosting the application. The application will use free softwares i.e, Tomcat server, PostgreSQL database which cuts off the expenses for hosting platform. There will be no additional cost for marketing as there will already be a complete website. Regarding the maintenance, the software development company will provide the free support for one year, in which system will be expected to be stable and error free.

3.1.3 Data Modeling (E-R Diagram)

An ER Diagram describes the structure of a database which is known as Entity Relationship Diagram. There are basically 3 things which we consider while making ER Diagram which are Entity, Relationship and Attributes. [9]

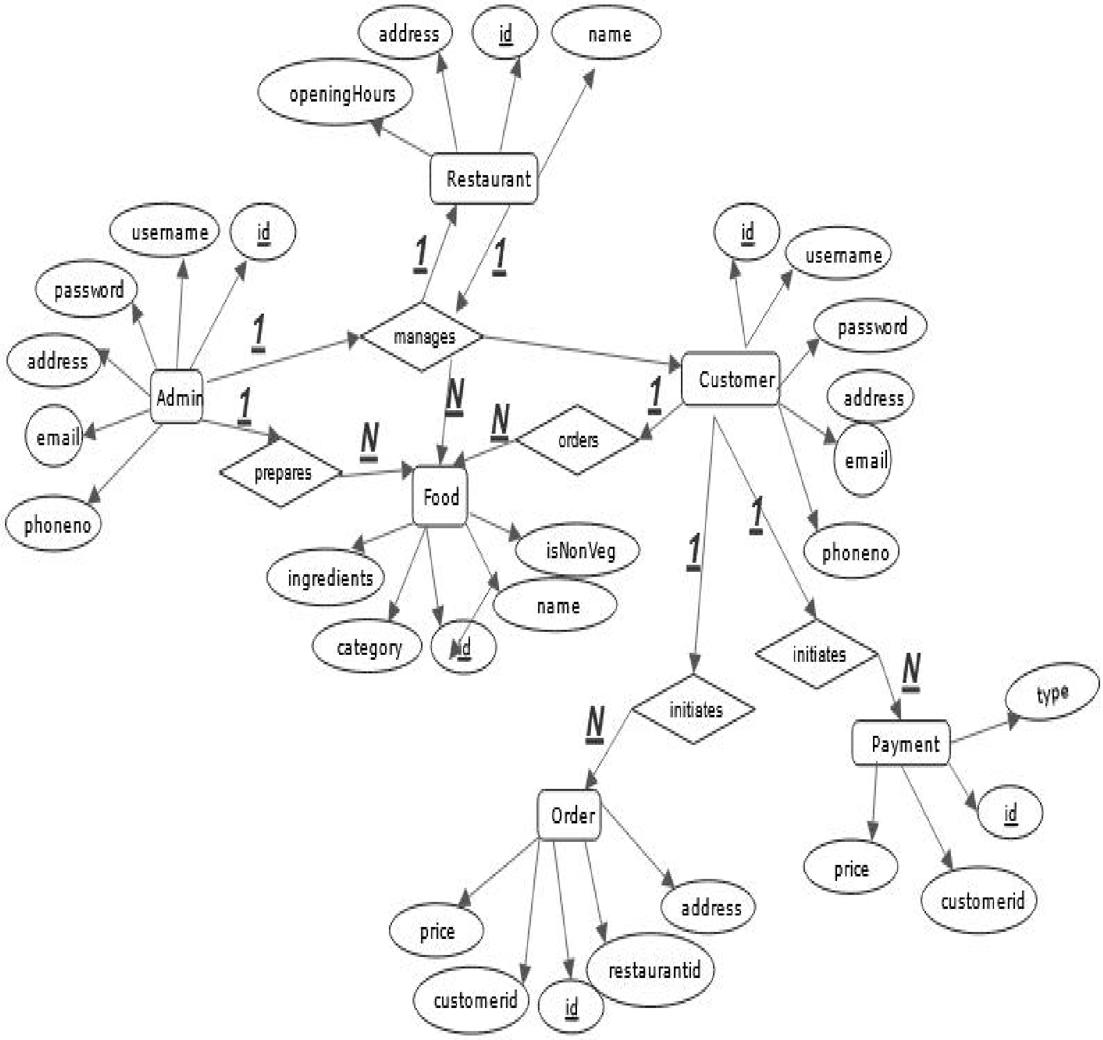


Figure 3.2: ER Diagram

In the Online Food Delivery System we have six tables (Admin, Customer, Restaurant, Food, Order, Payment) in the database which are here represented in the squares which represents entity where as its columns are in ovals which represents attributes as shown in the figure above. Their relationship is shown in the diamonds which represent relationships. Also, in the above figure, it is shown that whether they have one to one, one to many, many to one or many to many relationships with each other.

3.1.4 Process Modelling: DFD

Process modeling is the graphical representation of business processes or workflows. Like a flow chart, individual steps of the process are drawn out so there is an end-to- end overview of the tasks in the process within the context of the business environment. [10]



Figure 3.3: Zero Level DFD

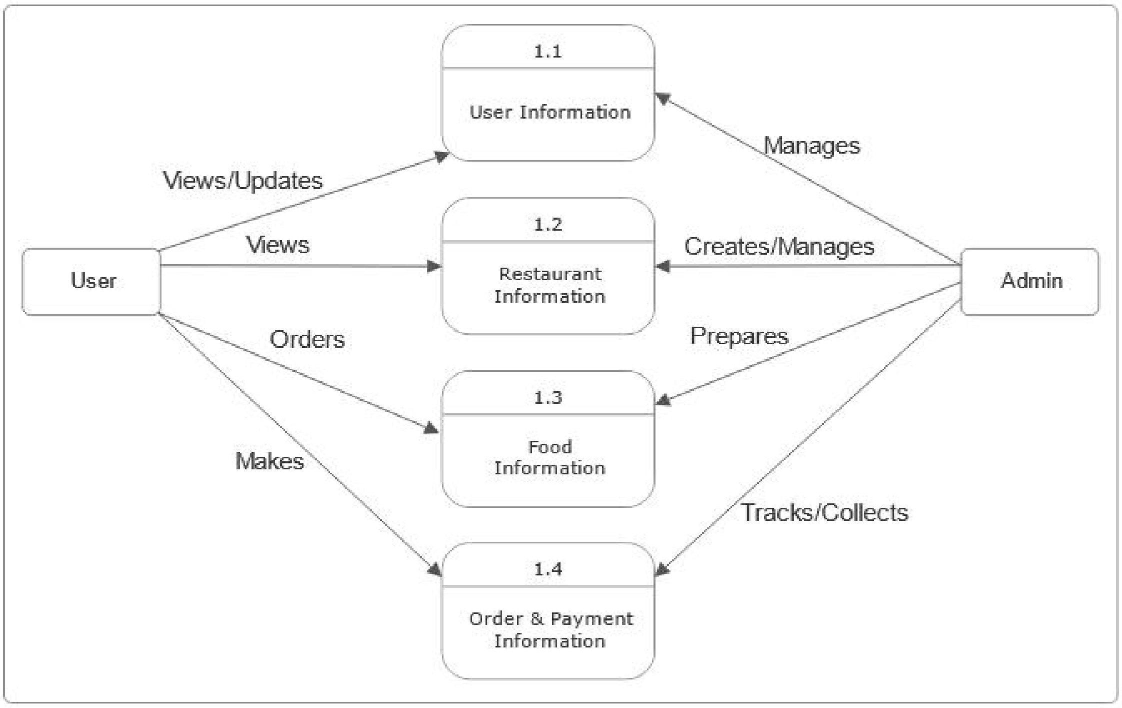


Figure 3.4: Level 1 DFD

Here, in the above figures the flow of data of Bite-Rush has been shown as in Two levels of DFD i.e. Level 0 and Level 1. There are two modules: user and admin. The admin can login to admin login and add restaurant, create food items and manages users orders and payment whereas the user can login to the system and add/view/update/delete their details and check restaurants and order food and pay online. The DFD omits technical details to focus on functional relationships, aligning with two-module design.

3.2 System Design

The architectural design, database schema design, interface UI design (UI/UX) and physical DFD of Bite-Rush are included here. System design is said to be the descriptive

in nature of what the system is and what is does and shows how the expected program is to be operated.

3.2.1 Architectural Design

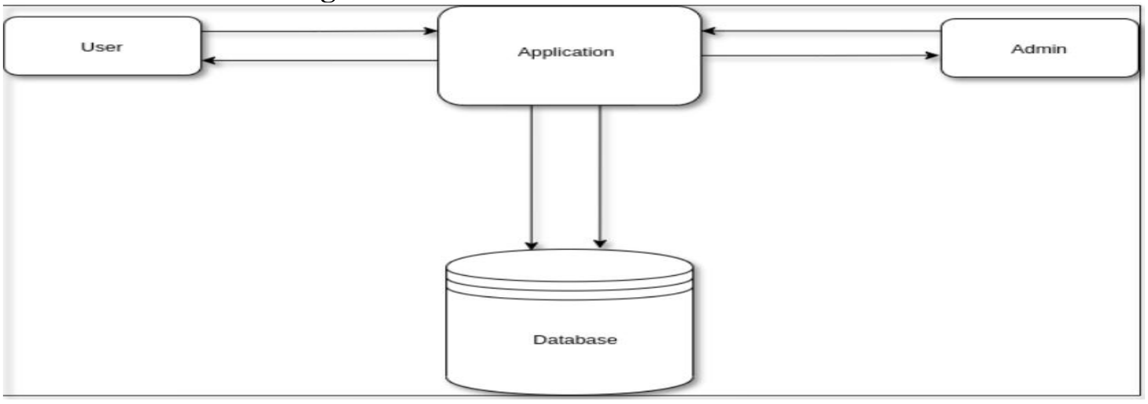


Figure 3.5: Architectural Diagram

3.2.2 Database Schema Design

Table "public.cart"

Column	Type
id	bigint
created_at	timestamp(6) without time zone
created_by_id	bigint
last_modified_at	timestamp(6) without time zone
modified_by_id	bigint
version	integer
cart_items	bigint[]
total	bigint
customer_id	bigint

Table "public.category"

Column	Type
id	bigint
created_at	timestamp(6) without time zone
created_by_id	bigint
last_modified_at	timestamp(6) without time zone
modified_by_id	bigint
version	integer
name	character varying(255)
restaurant_id	bigint

Table "public.orders"

Column	Type
id	bigint
created_at	timestamp(6) without time zone
created_by_id	bigint
last_modified_at	timestamp(6) without time zone
modified_by_id	bigint
version	integer
order_status	character varying(255)
total_items	integer
total_price	bigint
customer_id	bigint
delivery_address_id	bigint

Table "public.food"

Column	Type
id	bigint
created_at	timestamp(6) without time zone
created_by_id	bigint
last_modified_at	timestamp(6) without time zone
modified_by_id	bigint
version	integer
available	boolean
creation_date	timestamp(6) without time zone
description	character varying(255)
food_category_id	bigint
is_non_veg	boolean
is_seasonal	boolean
is_vegetarian	boolean
name	character varying(255)
price	bigint
restaurant_id	bigint

Table "public.restaurant"

Column	Type
id	bigint
created_at	timestamp(6) without time zone
created_by_id	bigint
last_modified_at	timestamp(6) without time zone
modified_by_id	bigint
version	integer
email	character varying(255)
facebook	character varying(255)
instagram	character varying(255)
linkedin	character varying(255)
mobile	character varying(255)
twitter	character varying(255)
cuisine_type	character varying(255)
description	character varying(255)
name	character varying(255)
open	boolean
opening_hours	character varying(255)
registration_date	timestamp(6) without time zone
address_id	bigint
owner_id	bigint

Table "public.users"

Column	Type
id	bigint
created_at	timestamp(6) without time zone
created_by_id	bigint
last_modified_at	timestamp(6) without time zone
modified_by_id	bigint
version	integer
email	character varying(255)
full_name	character varying(255)
password	character varying(255)
role	character varying(255)

Figure 3.6: Database Tables Schema

3.2.3 Interface Design

Interface design is a crucial aspect of creating an effective and user-friendly system. A well-designed interface can enhance the user experience, increase user engagement, and ultimately leads to higher customer satisfaction.

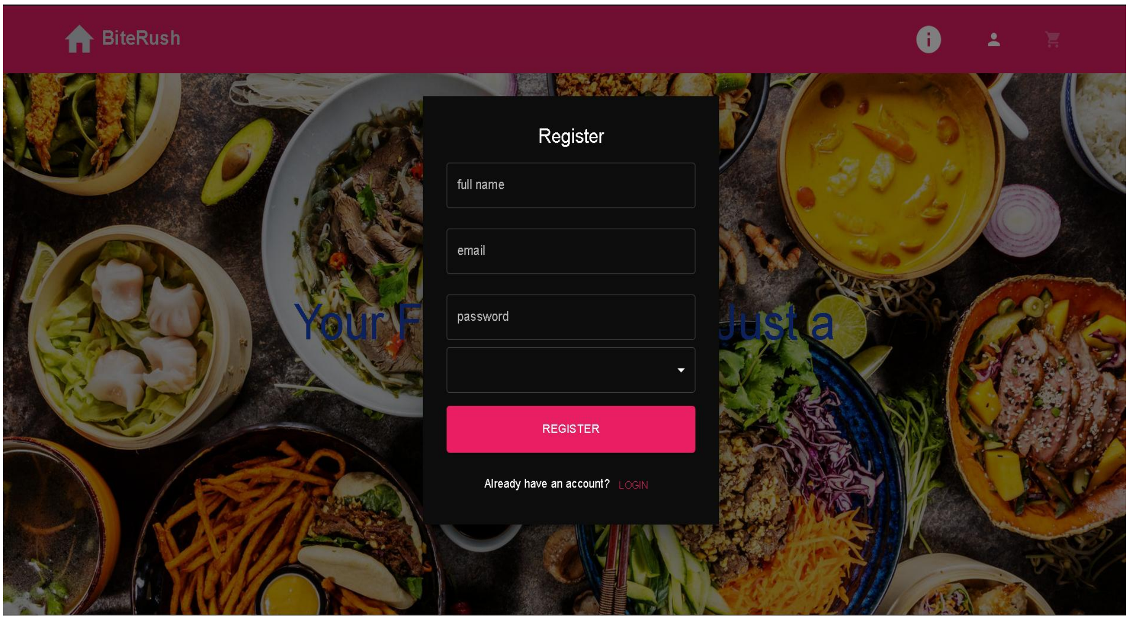


Figure 3.7: Registration Page

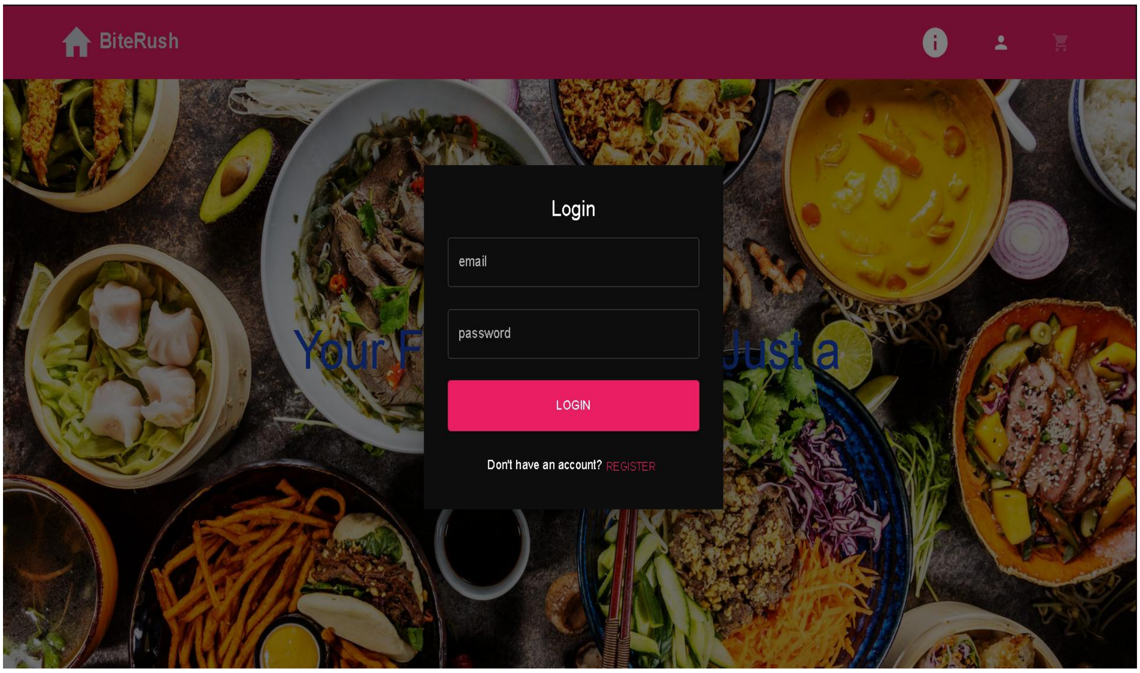


Figure 3.8: Login Page

This is the Food Delivery System's interface UI/UX design. The appealing UI design will definitely attract the attention and interest of the user in using the application.

3.2.4: Physical DFD

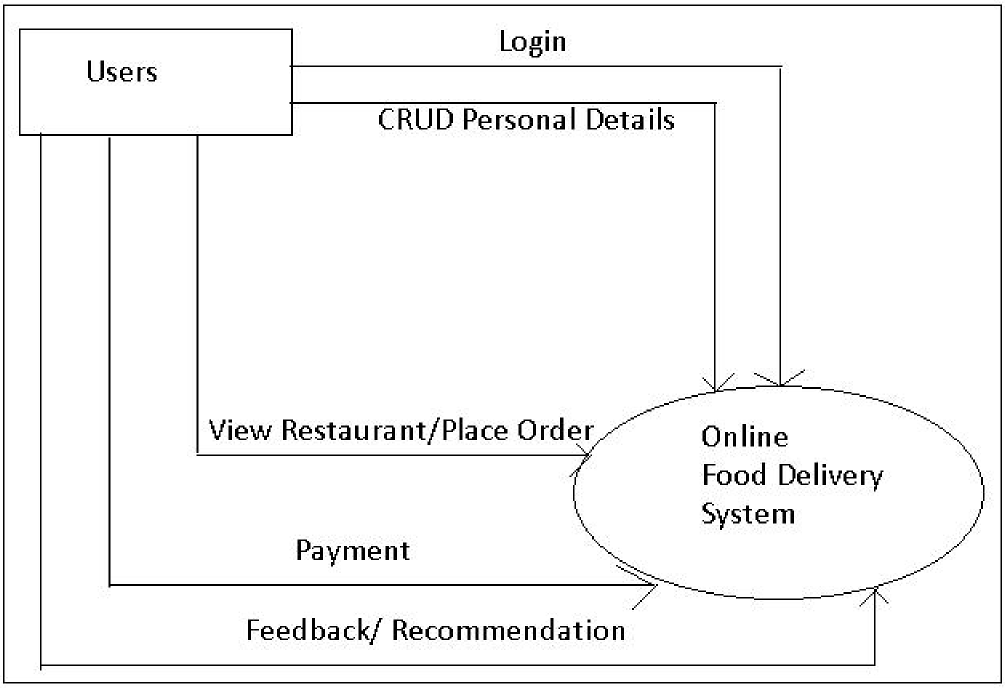


Figure 3.9: Physical DFD

A Physical Data Flow Diagram is a graphical representation that depicts the flow of data within a system at the implementation level. Unlike a Logical DFD, which focuses on the system's logical processes and data flows regardless of how they are physically implemented, a Physical DFD provides details about the actual implementation of the system, including hardware, software, databases, and external entities. [11]

CHAPTER IV: IMPLEMENTATION & TESTING

4.1 Implementation

4.1.1 Tools Used

Tools & Technology	Purposes
Java, SpringBoot	For backend development
PostgreSQL	For Database Management
React JS, Tailwind CSS	For Frontend UI Design and styling
Intellij Idea, VS Code	For coding and development
Git and GitHub	For version controlling

Table 4.1: Tools and Technologies

4.1.2: Implementation Details of Module

Implementation details provide a structured approach for developing the various modules of the Bite-Rush app, outlining the procedures and functions required for each module's functionality.

Customer Module:

1. User Registration/Login: This module facilitates user registration and login processes, ensuring secure access to the application.

Register user (username, password, email): Accepts user-provided username, password, and email, validates the input data, inserts the user information into the User Database rturns success or failure status, validates the input data, inserts the user information into the User Database and returns success or failure status.

Login user (username, password): accepts user-provided username and password , validates the credentials against the User Database, generates a session token upon successful authentication and returns the session token or error message.

2. Restaurant Discovery: Dynamic search with filters: Location, Cuisine, Price Range, and Ratings, Google Maps integration to show nearby restaurants, Favorites" feature to save frequently ordered restaurants.

3. Order Placement: Cart system with real-time price calculation (including taxes/delivery fees), Special instructions field for custom requests (e.g., "Less spicy", Order confirmation sent via SMS/email.

4. Esewa Payment Integration: Redirects to eSewa payment page with order details pre-filled, Automated payment verification via **eSewa's API** to confirm transactions, Fallback option for cash-on-delivery (if enabled).

5. Order Tracking: Real-time updates: Confirmed, Cooking, On the Way, Delivered, Delivery agent contact button for direct communication.

6: Rating & Reviews: Star ratings (1–5) and text reviews for restaurants, Photo upload option for food quality proof.

Restaurant Admin Module:

1. **Restaurant Onboarding:** Admin uploads: Menu (with categories like "Starters", "Main Course"), Photos, Pricing, and Delivery Zones., Approval system for new restaurants (optional super-admin role).

2. **Menu Management:** Bulk CSV upload for adding food items, Toggle to mark items as Available/Unavailable, Dynamic pricing for peak hours or discounts.

3. **Order Management:** Dashboard showing: New Orders, Preparing, Ready for Pickup. Accept/Reject orders with reason (e.g., "Out of stock"). Estimated preparation time calculator.

4. **Delivery Coordination:** Assign orders to delivery staff manually/automatically. Live location sharing with delivery personnel.

5. **Sales & Analytics:** Daily/weekly sales reports (exportable as PDF/Excel). Top-selling items and customer demographics.

6. **Customer Feedback:** View and respond to reviews from dashboard. Complaint resolution system for order issues.

4.2. Testing

Testing is evaluation of the software against requirements gathered from users and system specifications. Testing identifies important defects, flaws, or an error in the application code that must be fixed. It also assesses the features of a system. Testing assesses the quality of the product. The Online Food Delivery System was tested both individually and all together as agile methodology was used to develop the system.

4.2.1. Test Case for Unit Testing

Unit testing focuses on the modules independently locate the errors. This enables the tester to detect errors in coding. It is the process of taking a module and running it in isolation from rest of the software product by using prepared test cases and comparing the actual result with the result redirected with the specifications and design of the module. Since agile methodology was used to develop the system after each and every module was completed and was tested as a single unit.

Test case no.	Input given	Expected output	Actual output occurred	Test pass	Action taken
1	User: roshan Password: roshan	User login page or home page	User login page or home page	Yes	-
2	User: admin password: janjyoti	Admin home page	Invalid password for user admin	No	Wrong password janjyoti given for user admin
3	Adding new user according to user name	Added successfully	added	Yes	-

Table 4.2: Login/Signup Test Cases

Serial No.	Description	Expected Result	Actual Result	Result
1.	Check	User should be able to view the available restaurants	User views the available restaurants	Passed
2.	Order	User can place favorite food items in order	User adds items in cart and places for delivery	Passed
3.	View Delivery Details	User can get delivery details in real time	User easily tracks their order and delivery time	Passed
4.	Feedback	User can give reviews and suggestions	User shares his experience with the food	Passed

Table 4.3: Restaurant and User Review Test Cases

Serial No.	Description	Expected Result	Actual Result	Result
1.	Logout	Dashboard should close and login page should open.	Dashboard will close and login page will open.	Passed

Table 4.4: Logout Test Case

4.2.2 Test Case for System Testing

System Testing

Serial No.	Description	Expected Result	Actual Result	Result
1.	Overall system testing	All the functionalities of the system should work properly after integration.	All the functionalities of the system are working properly after integration.	Passed

Table 4.5: System Testing Test Case

CHAPTER V: CONCLUSION & FUTURE RECOMMENDATION

5.1. Conclusion

The **BITE-RUSH** Online Food Delivery System successfully addresses the gaps in Nepal's digital food industry by providing a localized, efficient, and user-friendly platform for customers and restaurants. By integrating eSewa payments, real-time order tracking, and a streamlined two-module system (Customer and Restaurant Admin), the project enhances convenience, transparency, and operational efficiency. The system empowers small restaurants with direct order management while offering customers a seamless experience—from browsing menus to secure payments and delivery updates. Future enhancements could include AI-driven recommendations, loyalty programs, and expanded payment options, further solidifying BITE-RUSH as a key player in Nepal's growing food-tech ecosystem. This project demonstrates how targeted digital solutions can transform traditional industries, benefiting businesses and consumers alike. [12]

5.2. Lesson Learnt/Outcome

The food delivery system is expected to streamline the process of ordering and delivering food from restaurants to customers efficiently. Upon completion, the project will provide a user-friendly web or mobile interface where users can browse restaurants, view menus, place orders, and track delivery in real-time.

The platform will facilitate:

- Seamless interaction between customers, delivery personnel, and restaurants
- Secure and reliable payment processing
- Order history and status tracking
- Restaurant management features for handling orders and menu updates

This system aims to reduce manual workload, improve delivery accuracy and speed, and enhance the overall user experience. Ultimately, it will create a reliable digital ecosystem that benefits both service providers and customers in the food industry.

5.3. Future Recommendation

To further enhance **BITE-RUSH**, the following improvements can be considered:

1. **Mobile App Development** – Expand accessibility with dedicated iOS and Android apps featuring push notifications for order updates and exclusive deals.
2. **AI-Powered Recommendations** – Implement machine learning to suggest dishes based on user preferences and order history.
3. **Multi-Payment Gateway Integration** – Add support for Khalti, Fonepay, and credit/debit cards to cater to diverse user preferences.
4. **Delivery Fleet Optimization** – Incorporate GPS route optimization and real-time tracking for faster, more efficient deliveries.
5. **Subscription & Loyalty Programs** – Introduce membership plans, discounts, and reward points to boost customer retention.
6. **Cloud Kitchen Partnerships** – Collaborate with virtual restaurants to expand food options without physical store requirements.
7. **Multi-Language Support** – Add Nepali and Newari language options for broader user engagement.

References

[1] "Online food delivery trends in Nepal," Foodmandu, 2023. [Online]. Available: https://www.foodmandu.com. [Accessed: 15-Jan-2024].

[2] "eSewa Payment Gateway API documentation," eSewa, 2023. [Online]. Available: https://developer.esewa.com.np. [Accessed: 20-Jan-2024].

[3] "Digital payment adoption in Nepal," Nepal Rastra Bank, 2023. [Online]. Available: https://www.nrb.org.np. [Accessed: 25-Jan-2024].

[4] "MongoDB for food delivery applications," MongoDB Inc., 2023. [Online]. Available: https://www.mongodb.com/use-cases/food-delivery. [Accessed: 30-Jan-2024].

[5] "React.js documentation for web applications," Facebook, 2023. [Online]. Available: https://reactjs.org/docs. [Accessed: 5-Feb-2024].

[6] "Node.js best practices for REST APIs," Node.js Foundation, 2023. [Online]. Available: https://nodejs.org/en/docs/guides. [Accessed: 10-Feb-2024].

[7] "Real-time features with Socket.io," Socket.io, 2023. [Online]. Available: https://socket.io/docs/v4. [Accessed: 15-Feb-2024].

[8] "W3Schools web development tutorials," W3Schools, 2023. [Online]. Available: https://www.w3schools.com. [Accessed: 20-Feb-2024].

[9] "Stack Overflow developer community," Stack Overflow, 2023. [Online]. Available: https://stackoverflow.com. [Accessed: 25-Feb-2024].

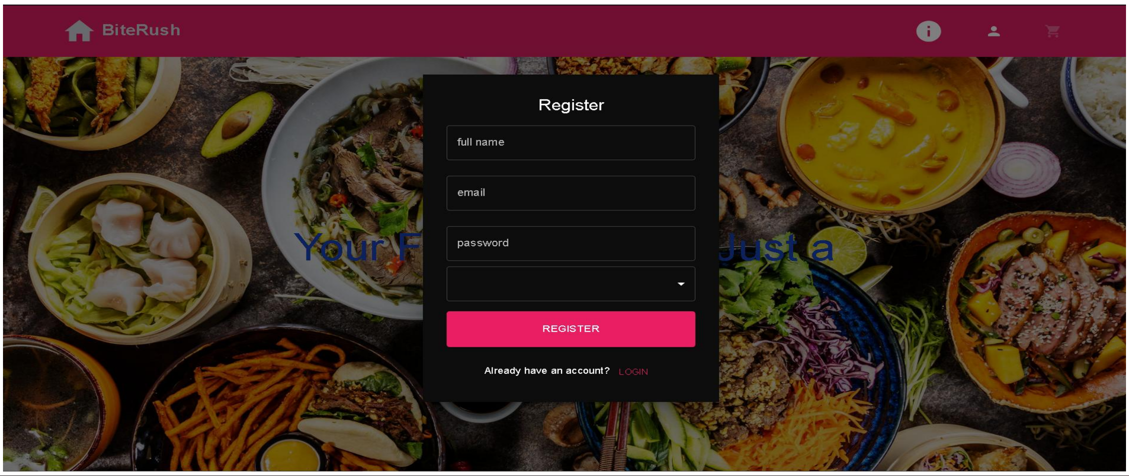
[10] "Digital Nepal Framework," Government of Nepal, 2023. [Online]. Available: https://www.digitalnepal.gov.np. [Accessed: 1-Mar-2024].

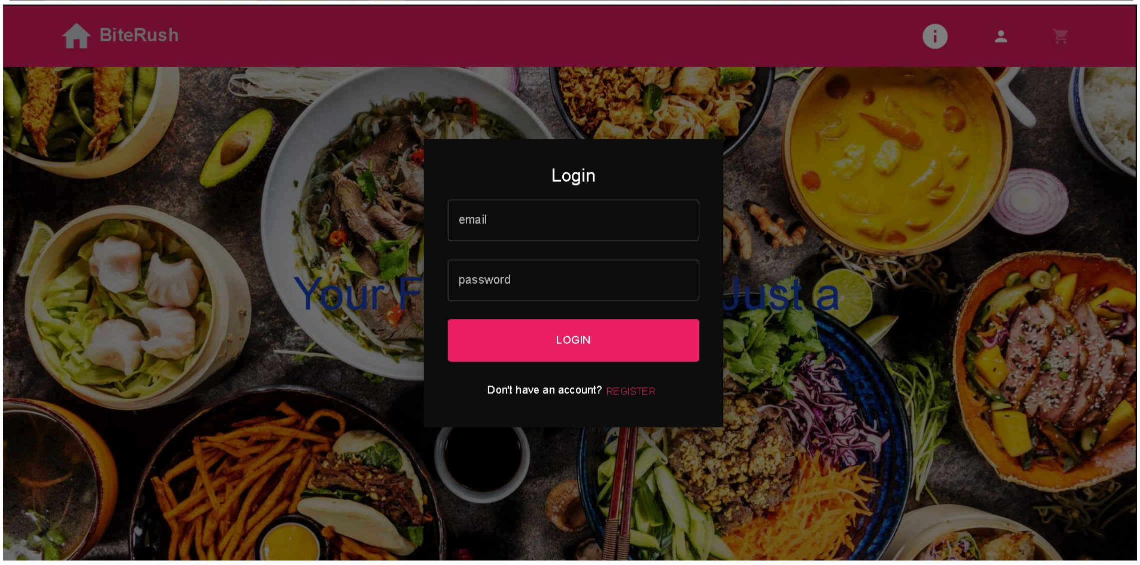
[11] "Nepal internet penetration statistics," Nepal Telecommunications Authority, 2023. [Online]. Available: https://nta.gov.np. [Accessed: 5-Mar-2024].

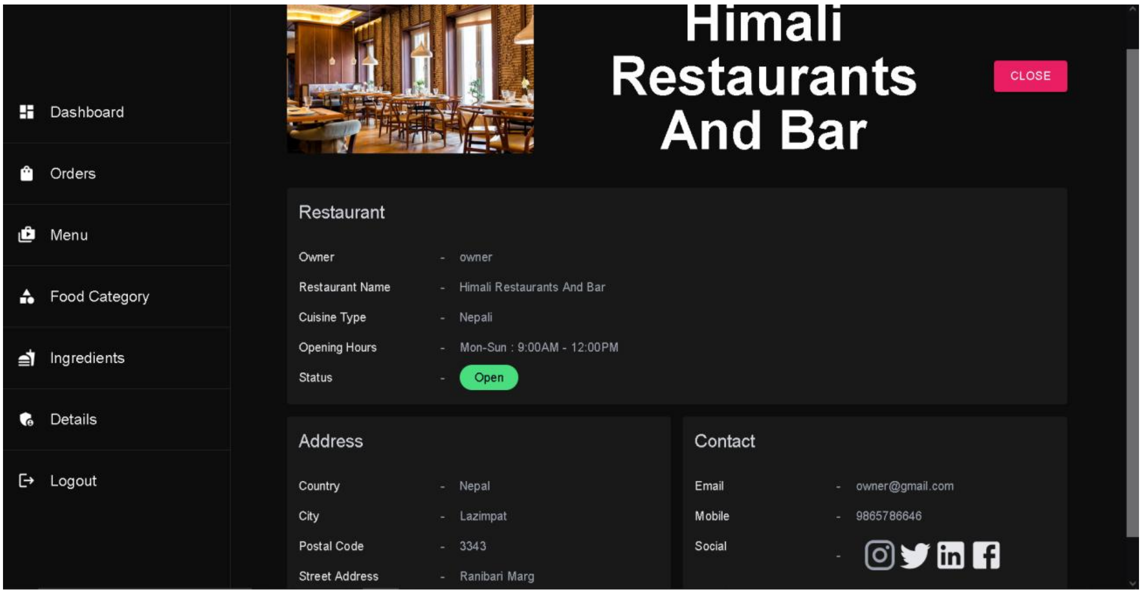
[12] "Khalti payment integration," Khalti, 2023. [Online]. Available: https://developer.khalti.com. [Accessed: 10-Mar-2024

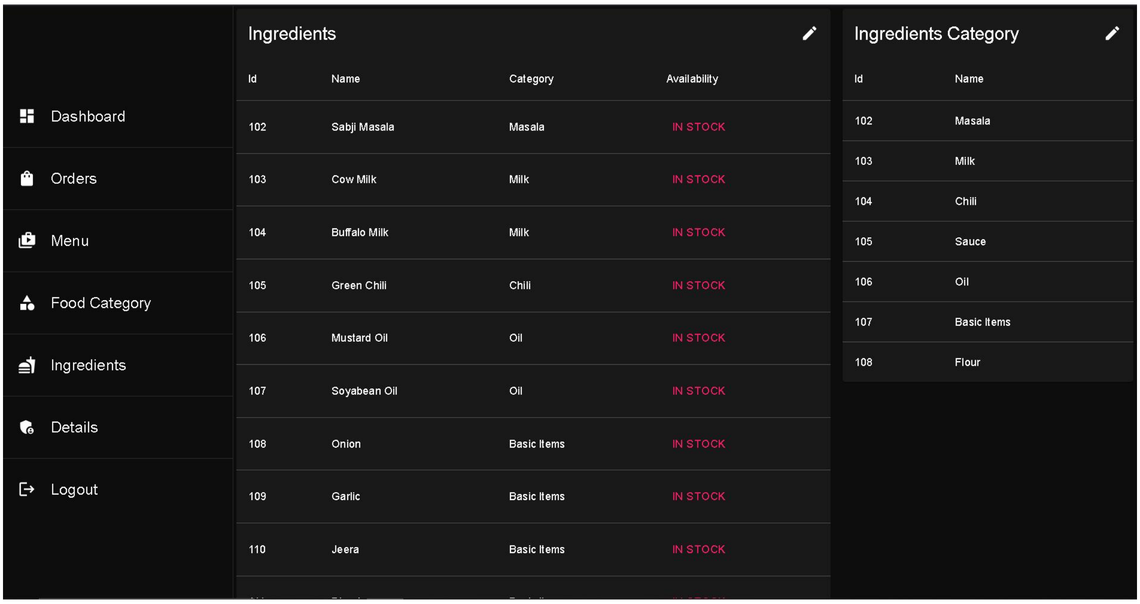
Appendices

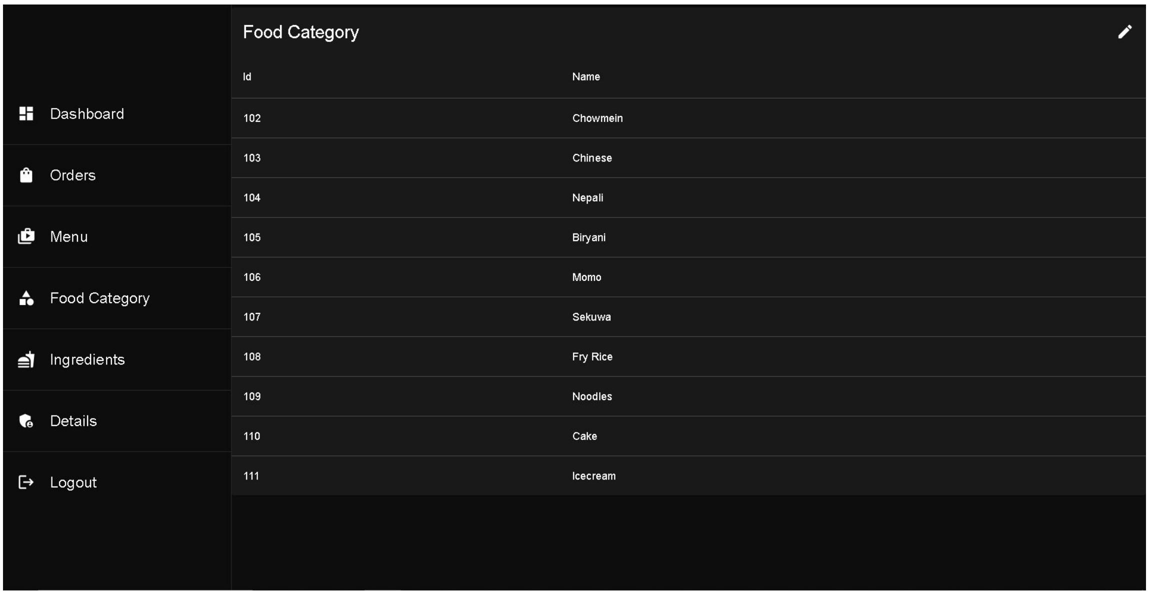
Screenshots:

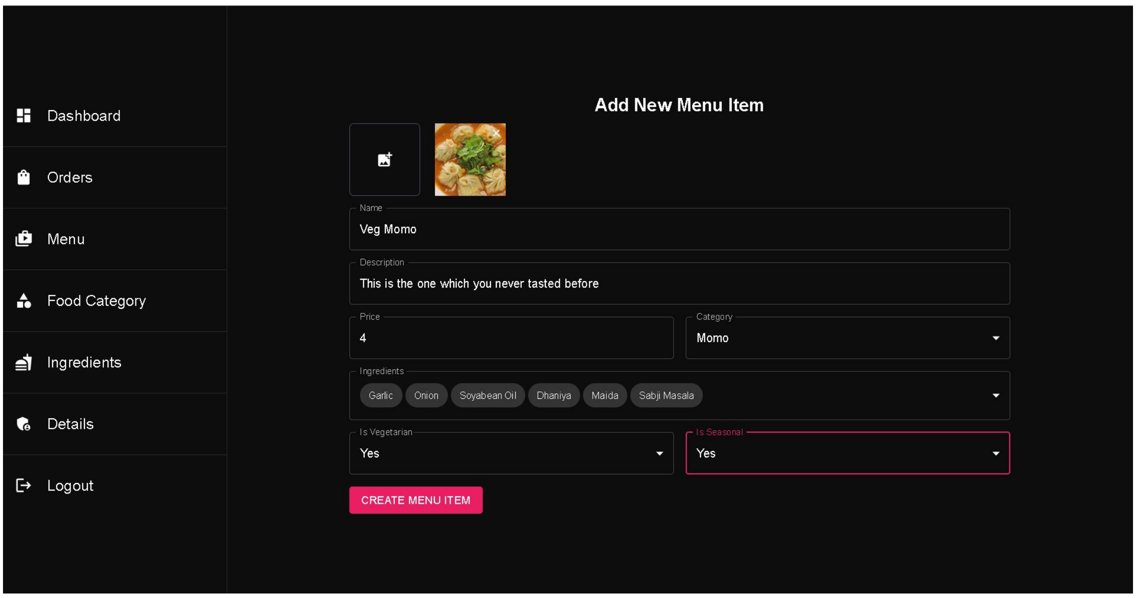


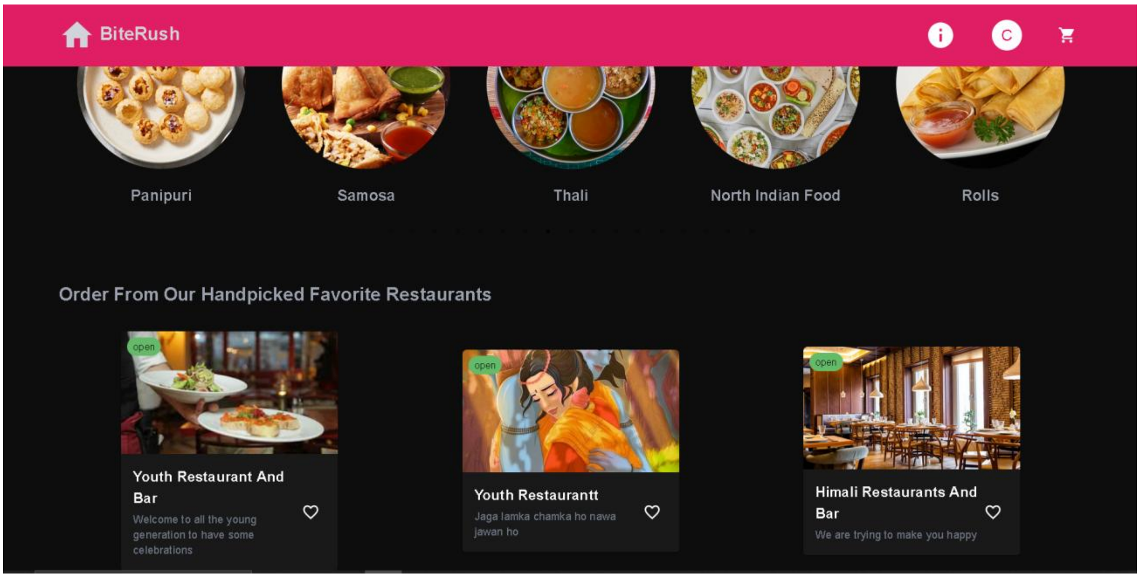


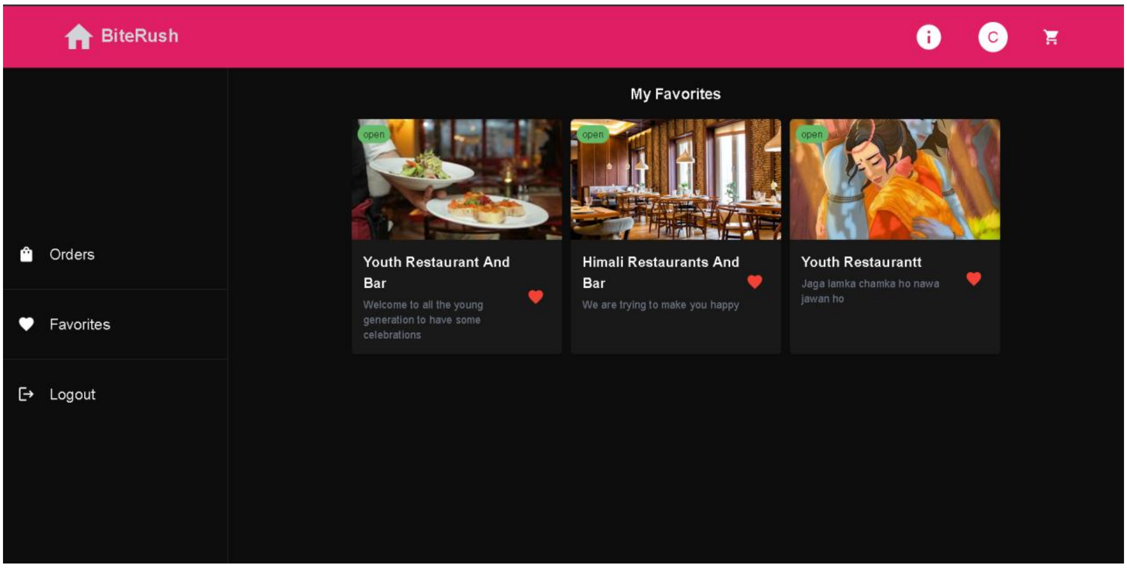


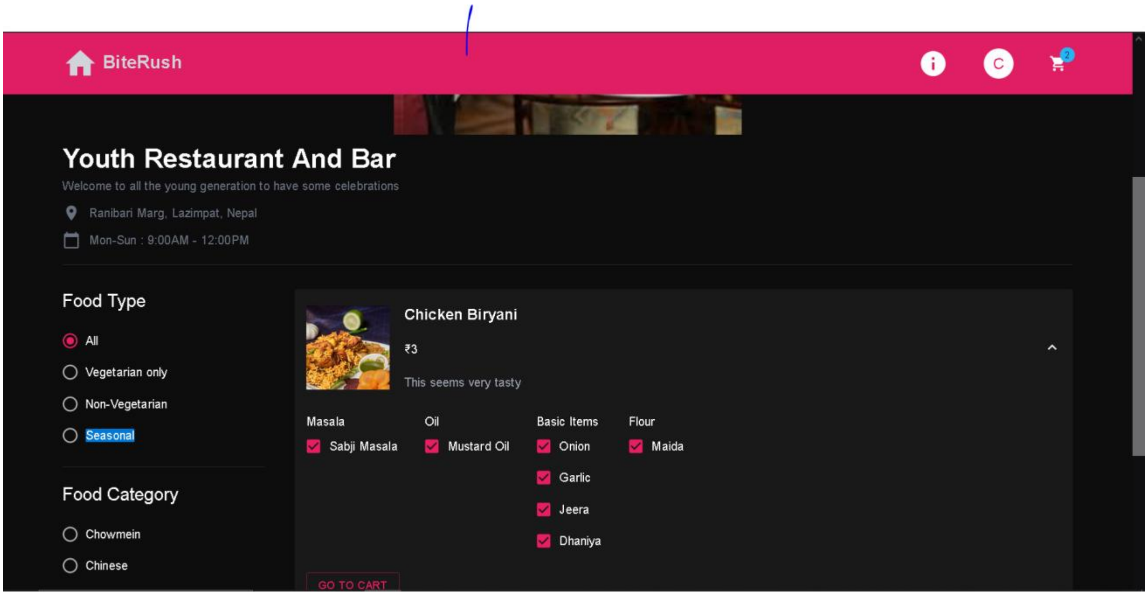

Dashboard
Orders
Menu
Food Category
Ingredients
Details
Logout
Ingredients
Id
Name
Category
Availability
102
Sabji Masala
Masala
IN STOCK
103
Cow Milk
Milk
IN STOCK
104
Buffalo Milk
Milk
IN STOCK
105
Green Chili
Chili
IN STOCK
106
Mustard Oil
Oil
IN STOCK
107
Soyabean Oil
Oil
IN STOCK
108
Onion
Basic Items
IN STOCK
109
Garlic
Basic Items
IN STOCK
110
Jeera
Basic Items
IN STOCK
Ingredients Category
Id
Name
102
Masala
103
Milk
104
Chili
105
Sauce
106
Oil
107
Basic Items
108
Flour

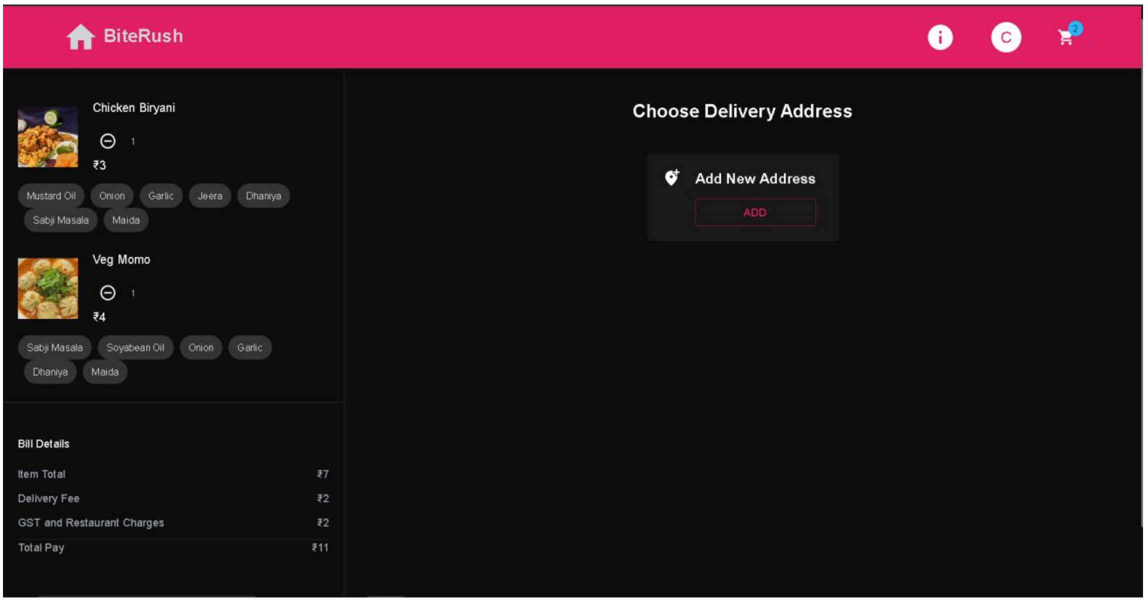

Dashboard
Orders
Menu
Food Category
Ingredients
Details
Logout
Food Category
Id
Name
102
Chowmein
103
Chinese
104
Nepali
105
Biryani
106
Momo
107
Sekuwa
108
Fry Rice
109
Noodles
110
Cake
111
Icecream

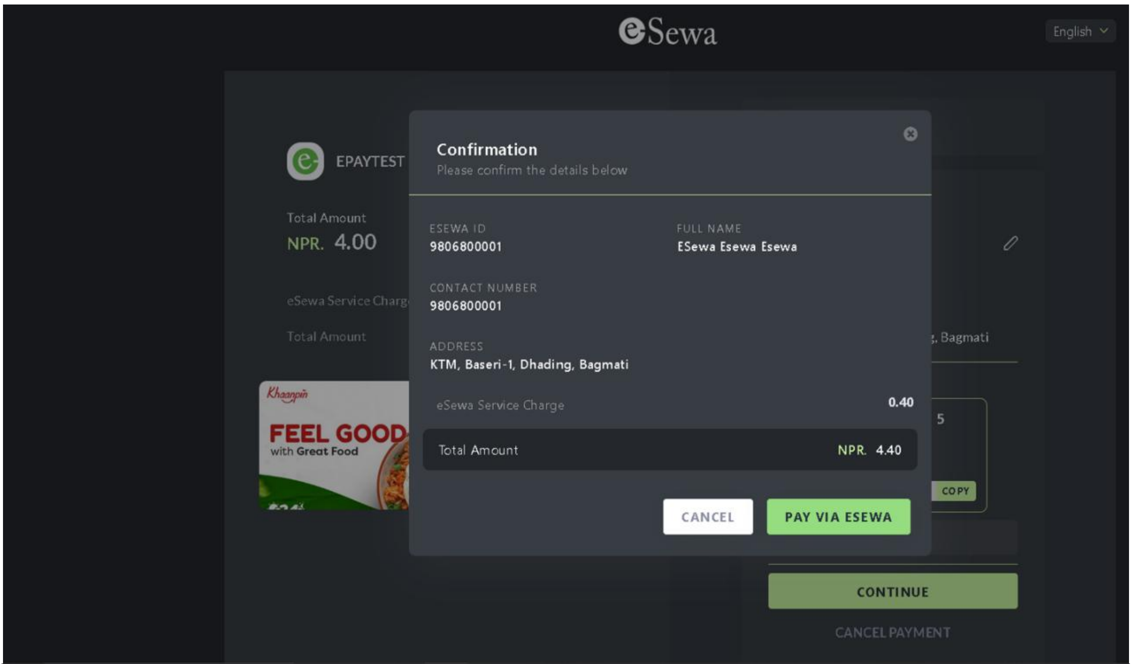

Dashboard
Orders
Menu
Food Category
Ingredients
Details
Logout
Add New Menu Item
Name
Veg Momo
Description
This is the one which you never tasted before
Price
4
Category
Momo
Ingredients
Garlic
Onion
Soyabean Oil
Dhaniya
Maida
Sabji Masala
Is Vegetarian
Yes
Is Seasonal
Yes
CREATE MENU ITEM

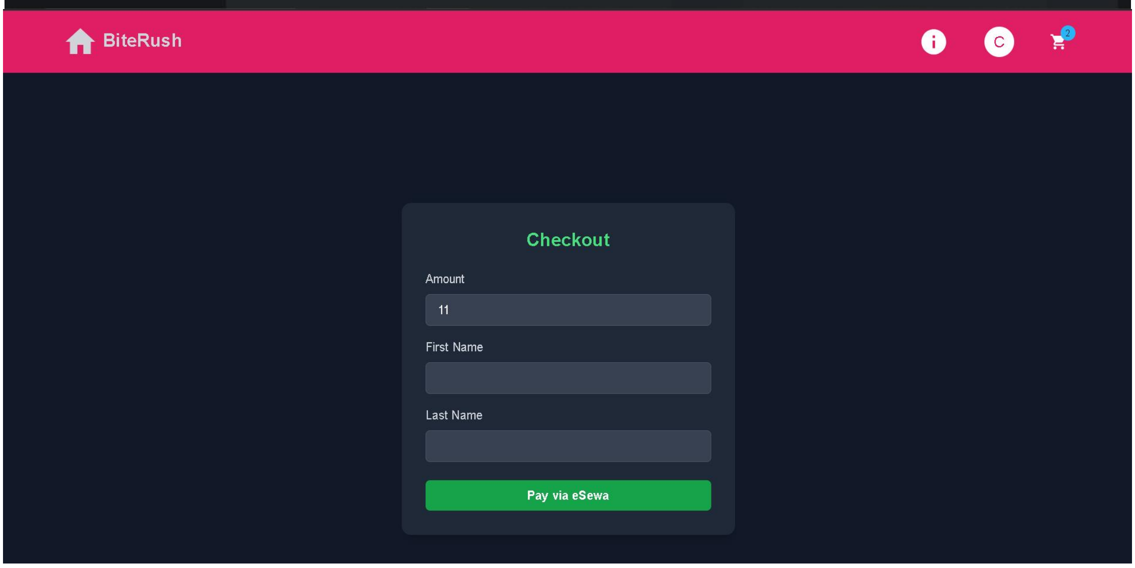

BiteRush
C
Panipuri
Samosa
Thali
North Indian Food
Rolls
Order From Our Handpicked Favorite Restaurants
open
Youth Restaurant And Bar
Welcome to all the young generation to have some celebrations
open
Youth Restaurantt
Jaga lamka chamka ho nawa jawan ho
open
Himali Restaurants And Bar
We are trying to make you happy


BiteRush
C
My Favorites
Orders
Favorites
Logout
open
Youth Restaurant And Bar
Welcome to all the young generation to have some celebrations
open
Himali Restaurants And Bar
We are trying to make you happy
open
Youth Restaurantt
Jaga lamka chamka ho nawa jawan ho


BiteRush
C
Youth Restaurant And Bar
Welcome to all the young generation to have some celebrations
Ranibari Marg, Lazimpat, Nepal
Mon-Sun : 9:00AM - 12:00PM
Food Type
All
Vegetarian only
Non-Vegetarian
Seasonal
Food Category
Chowmein
Chinese
Chicken Biryani
₹3
This seems very tasty
Masala
Sabji Masala
Oil
Mustard Oil
Basic Items
Onion
Garlic
Jeera
Dhaniya
Flour
Maida
GO TO CART


BiteRush
Chicken Biryani
1
₹3
Mustard Oil
Onion
Garlic
Jeera
Dhaniya
Sabji Masala
Maida
Veg Momo
1
₹4
Sabji Masala
Soyabean Oil
Onion
Garlic
Dhaniya
Maida
Bill Details
Item Total ₹7
Delivery Fee ₹2
GST and Restaurant Charges ₹2
Total Pay ₹11
Choose Delivery Address
Add New Address
ADD


eSewa
English
EPAYTEST
Total Amount
NPR. 4.00
eSewa Service Charg
Total Amount
FEEL GOOD
with Great Food
Confirmation
Please confirm the details below
ESEWA ID
9806800001
FULL NAME
ESewa Esewa Esewa
CONTACT NUMBER
9806800001
ADDRESS
KTM, Baseri-1, Dhading, Bagmati
eSewa Service Charge 0.40
Total Amount NPR. 4.40
CANCEL
PAY VIA ESEWA
COPY
CONTINUE
CANCEL PAYMENT


BiteRush
Checkout
Amount
11
First Name
Last Name
Pay via eSewa

Source code:

UserProfile.jsx

src > components > Profile > UserProfile.jsx > UserProfile

```jsx
import React from 'react'
import AccountCircleIcon from '@mui/icons-material/AccountCircle';
import { Button } from '@mui/material';
import { useNavigate } from 'react-router-dom';
import { useDispatch, useSelector } from 'react-redux';
import { logout } from '../../State/Authentication/Action';

const UserProfile = () => {
  const navigate = useNavigate();
  const dispatch = useDispatch();
  const user = useSelector(store => store.auth.user);
  console.log('user profile user ', user);
  const handleLogout = ()=> {
    dispatch(logout(navigate));
    // navigate("/");
  }
  return (
    <div className='min-h-[80vh] flex flex-col justify-center items-center text-center'>
      <div className='flex flex-col items-center justify-center'>
        <AccountCircleIcon sx={{fontSize:'9rem'}} />
        <h1 className='py-5 text-2xl font-semibold'>{user.fullName}</h1>
        <p>{user.email}</p>
        <Button variant='contained' onClick={handleLogout} sx={{margin:'2rem 0rem'}}>Logout</Button>
      </div>
    </div>
  )
}

export default UserProfile
```

Payment.jsx

src > components > PaymentSuccess > Payment.jsx > ...

```jsx
import { useLocation } from 'react-router-dom';
import { useState, useEffect } from 'react';
import { v4 as uuidv4 } from 'uuid';
import CryptoJS from 'crypto-js';

const Payment = () => {
  const location = useLocation();
  const totalAmount = location.state?.totalAmount || 0;

  const [formData, setFormData] = useState({
    amount: totalAmount,
    tax_amount: '0',
    total_amount: totalAmount,
    transaction_uuid: uuidv4(),
    product_service_charge: '0',
    product_delivery_charge: '0',
    product_code: 'EPAYTEST',
    success_url: 'http://localhost:5173/payment/success',
    failure_url: 'http://localhost:5173/payment/failure',
    signed_field_names: 'total_amount,transaction_uuid,product_code',
    signature: '',
    secret: '8gBm/:&EnhH.1/q',
  });

  const generateSignature = (total_amount, transaction_uuid, product_code, secret) => {
    const hashString = `total_amount=${total_amount},transaction_uuid=${transaction_uuid},product_code=${product_code}`;
    const hash = CryptoJS.HmacSHA256(hashString, secret);
    return CryptoJS.enc.Base64.stringify(hash);
  };

  useEffect(() => {
    const { total_amount, transaction_uuid, product_code, secret } = formData;
    const hashedSignature = generateSignature(total_amount, transaction_uuid, product_code, secret);
    setFormData((prev) => ({ ...prev, signature: hashedSignature }));
  }, [formData.amount]);
```

RestaurantDetails.jsx

src > components > Restaurant > RestaurantDetails.jsx > ...

```jsx

const RestaurantDetails = () => {
  const { auth, restaurant, menu } = useSelector(store => store);
  const [foodType, setFoodType] = useState('all');
  const navigate = useNavigate();
  const dispatch = useDispatch();
  const jwt = auth.jwt || localStorage.getItem("jwtoriginal");
  const [ selectedCategory, setSelectedCategory ] = useState(null);

  const { id, city } = useParams();

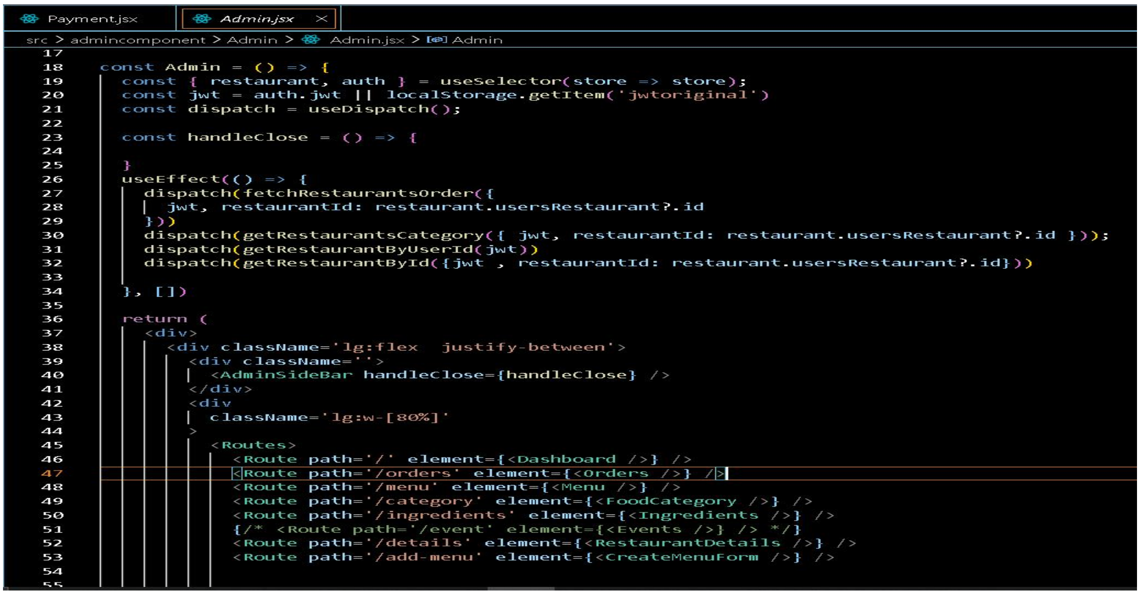
  const handleFilter = (e) => {
    setFoodType(e.target.value);
    console.log(e.target.value, e.target.name);
  }


  const handleFilterCategory = (e, value) => {
    setSelectedCategory(value);
    console.log(e.target.value, e.target.name, value);
  }

  useEffect(() => {
    dispatch(getRestaurantById({ jwt, restaurantId: id }));
    dispatch(getRestaurantsCategory({ jwt, restaurantId: id }));
  }, []);



  useEffect(() => {
  dispatch(getMenuItemsByRestaurantId({
    jwt,
    restaurantId: id,
    vegetarian: foodType === 'vegetarian',
    nonveg: foodType === 'non_vegetarian',
```



Payment.jsx | Admin.jsx

src > admincomponent > Admin > Admin.jsx > Admin

```jsx
const Admin = () => {
  const { restaurant, auth } = useSelector(store => store);
  const jwt = auth.jwt || localStorage.getItem('jwtoriginal')
  const dispatch = useDispatch();

  const handleClose = () => {

  }
  useEffect(() => {
    dispatch(fetchRestaurantsOrder({
      jwt, restaurantId: restaurant.usersRestaurant?.id
    }))
    dispatch(getRestaurantsCategory({ jwt, restaurantId: restaurant.usersRestaurant?.id }));
    dispatch(getRestaurantByUserId(jwt))
    dispatch(getRestaurantById({jwt , restaurantId: restaurant.usersRestaurant?.id}))

  }, [])

  return (
    <div>
      <div className='lg:flex  justify-between'>
        <div className=''>
          <AdminSideBar handleClose={handleClose} />
        </div>
        <div
          className='lg:w-[80%]'
        >
          <Routes>
            <Route path='/' element={<Dashboard />} />
            <Route path='/orders' element={<Orders />} />
            <Route path='/menu' element={<Menu />} />
            <Route path='/category' element={<FoodCategory />} />
            <Route path='/ingredients' element={<Ingredients />} />
            {/* <Route path='/event' element={<Events />} /> */}
            <Route path='/details' element={<RestaurantDetails />} />
            <Route path='/add-menu' element={<CreateMenuForm />} />
```



Action.js

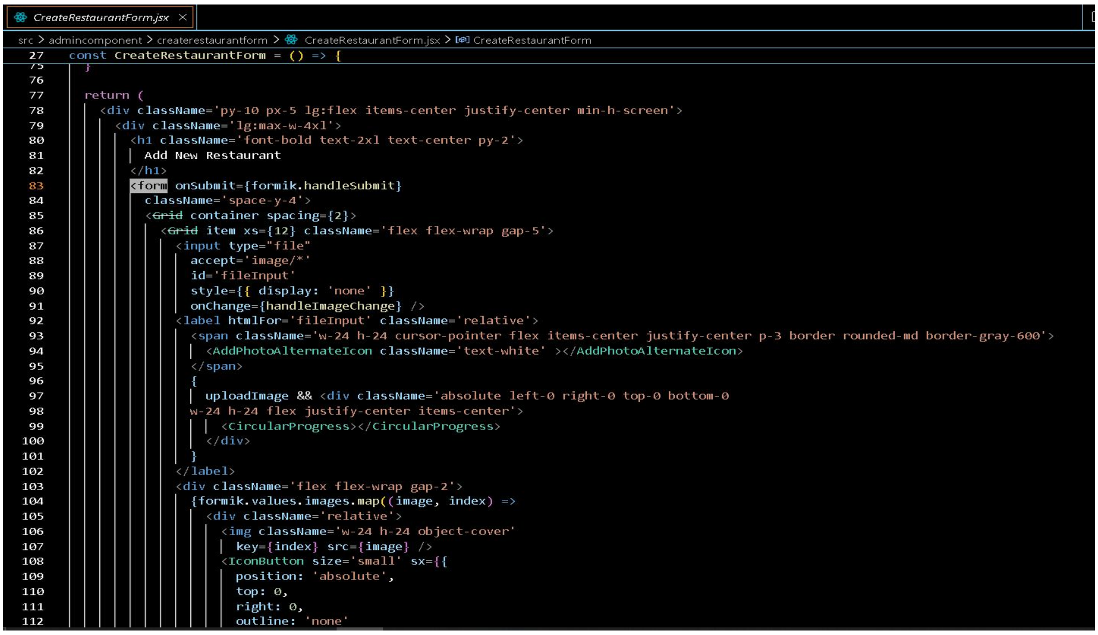
src > State > Authentication > Action.js > logout > <function> > type

```js
export const registerUser = (reqData) => async (dispatch) => {
  dispatch({ type: REGISTER_REQUEST });
  try {
    const { data } = await axios.post(`${API_URL}/auth/signup`, reqData.userData);
    if (data.jwt) {
      localStorage.setItem('jwtoriginal', data.jwt);
      dispatch({ type: REGISTER_SUCCESS, payload: data.jwt });

      if (data.role === 'ROLE_RESTAURANT_OWNER') {
        reqData.navigate('/admin/restaurant');
      } else {
        reqData.navigate('/');
      }
    }
    console.log('register success', data);
  } catch (error) {
    dispatch({ type: REGISTER_FAILURE, payload: error });
    console.log("error", error);
  }
};


export const loginUser = (reqData) => async (dispatch) => {
  dispatch({ type: LOGIN_REQUEST });
  try {
    const { data } = await api.post(`${API_URL}/auth/signin`, reqData.userData);
    const jwt = data.jwt;
    localStorage.setItem('jwtoriginal', jwt);
    console.log('jwt after logging in ', jwt);
    if (data.jwt) {
      dispatch({ type: LOGIN_SUCCESS, payload: data.jwt });
      await dispatch(getUser(jwt));
      await dispatch(getAllRestaurantsAction(jwt));

      if (data.role === 'ROLE_RESTAURANT_OWNER') {
        reqData.navigate('/admin/restaurant');
      } else {
        reqData.navigate('/');
```



CreateRestaurantForm.jsx

src > admincomponent > createrestaurantform > CreateRestaurantForm.jsx > CreateRestaurantForm

```jsx
const CreateRestaurantForm = () => {
  }

  return (
    <div className='py-10 px-5 lg:flex items-center justify-center min-h-screen'>
      <div className='lg:max-w-4xl'>
        <h1 className='font-bold text-2xl text-center py-2'>
          Add New Restaurant
        </h1>
        <form onSubmit={formik.handleSubmit}
          className='space-y-4'>
          <Grid container spacing={2}>
            <Grid item xs={12} className='flex flex-wrap gap-5'>
              <input type="file"
                accept='image/*'
                id='fileInput'
                style={{ display: 'none' }}
                onChange={handleImageChange} />
              <label htmlFor='fileInput' className='relative'>
                <span className='w-24 h-24 cursor-pointer flex items-center justify-center p-3 border rounded-md border-gray-600'>
                  <AddPhotoAlternateIcon className='text-white' ></AddPhotoAlternateIcon>
                </span>
                {
                  uploadImage && <div className='absolute left-0 right-0 top-0 bottom-0
                w-24 h-24 flex justify-center items-center'>
                    <CircularProgress></CircularProgress>
                  </div>
                }
              </label>
              <div className='flex flex-wrap gap-2'>
                {formik.values.images.map((image, index) =>
                  <div className='relative'>
                    <img className='w-24 h-24 object-cover'
                      key={index} src={image} />
                    <IconButton size='small' sx={{
                      position: 'absolute',
                      top: 0,
                      right: 0,
                      outline: 'none'
```

FoodServiceImpl.java

```java
public class FoodServiceImpl implements IFoodService {

    @Override
    public Food createFood(CreateFoodRequest req, Long categoryId, Restaurant restaurant) {
        Food food = new Food();
        Optional<Category> byId = categoryRepo.findById(categoryId);
        food.setFoodCategoryId(byId.get().getId());
        food.setRestaurantId(restaurant.getId());
        food.setDescription(req.getDescription());
        food.setImages(req.getImages());
        food.setName(req.getName());
        food.setPrice(req.getPrice());
        for(Long id: req.getIngredientsIds()) {
            Ingredient i = ingredientRepo.findById(id).orElse( other: null);
            if(i != null) {
                food.getIngredients().add(i);
            } else {
                throw new RuntimeException("Ingredient with id : " + id + " does not exist!");
            }
        }
        food.setSeasonal(req.isSeasonal());
        food.setVegetarian(req.isVegetarian());
        food.setNonVeg(req.isNonVeg());
        food.setCreationDate(new Date());
        Food savedFood = foodRepo.save(food);
//        restaurant.getFoods().add(savedFood);
        return savedFood;
    }

    @Override
    public void deleteFood(Long foodId) throws Exception {
        Food food = foodRepo.findById(foodId).orElseThrow(() -> new RuntimeException("Food not found"));
        food.getImages().clear();
        foodRepo.delete(food);
```

UserController.java

```java
import java.util.ArrayList;
import java.util.List;

@RestController
@RequestMapping("/users")
@RequiredArgsConstructor
public class UserController {

    private final IUserService userService;

    private final IFavoriteRestaurantRepo favoriteRestaurantRepo;
    private final IRestaurantRepo restaurantRepo;;

    @GetMapping("/profile")
    public ResponseEntity<Users> findUserByJwtToken(@RequestHeader("Authorization") String jwt) throws Exception {
        Users user = userService.findUserByJwtToken(jwt);
        return new ResponseEntity<>(user, HttpStatus.OK);
    }

    @GetMapping("/favorites")
    public ResponseEntity<List<Restaurant>> getAllUsersFavorites(@RequestHeader("Authorization") String jwt) throws Exception {
        Users user = userService.findUserByJwtToken(jwt);
        List<FavoriteRestaurant> fav = favoriteRestaurantRepo.findByUserId(user.getId());
        List<Restaurant> restaurants = new ArrayList<>();

        for (FavoriteRestaurant fr : fav) {
            restaurantRepo.findById(fr.getRestaurantId())
                    .ifPresent(restaurants::add);
        }
        return new ResponseEntity<>(restaurants, HttpStatus.OK);
    }
```

AdminFoodController.java

```java
public class AdminFoodController {
    private final IFoodService foodService;
    private final IRestaurantService restaurantService;

    @PostMapping("/create")
    public ResponseEntity<Food> createFood(@RequestBody CreateFoodRequest req) throws Exception {
        Restaurant restaurant = restaurantService.findRestaurantById(req.getRestaurantId());
        if(!req.isVegetarian()) {
            req.setNonVeg(true);
        }
        Food food = foodService.createFood(req, req.getCategoryId(), restaurant);
        return new ResponseEntity<>(food, HttpStatus.CREATED);
    }

    @DeleteMapping("/delete/{id}")
    public ResponseEntity<MessageResponse> DeleteFood(@PathVariable Long id) throws Exception {
        foodService.deleteFood(id);
        MessageResponse response = new MessageResponse();
        response.setMessage("Food deleted successfully...");
        return new ResponseEntity<>(response, HttpStatus.OK);
    }

    @PutMapping("/status/{id}")
    public ResponseEntity<Food> updateFoodAvailabilityStatus(@PathVariable Long id) throws Exception {
        Food food = foodService.updateAvailabilityStatus(id);
        return new ResponseEntity<>(food, HttpStatus.OK);
    }
}
```

JwtTokenValidator.java | JwtProvider.java

```java
@Service
public class JwtProvider {

    private final SecretKey key = Keys.hmacShaKeyFor(JwtConstant.SECRET_KEY.getBytes());

    public String generateToken(Authentication auth) {
        Collection<? extends GrantedAuthority> authorities = auth.getAuthorities();
        String roles = populateAuthorities(authorities);
        return Jwts.builder()
                .issuedAt(new Date()).expiration(new Date(new Date().getTime() + 86400000))
                .claim( s: "email", auth.getName()).claim( s: "authorities", roles)
                .signWith(key).compact();
    }

    public String getEmailFromJwtToken(String jwt) {
        Claims claims = Jwts.parser().verifyWith(key).build().parseSignedClaims(jwt).getPayload();
        return String.valueOf(claims.get("email"));
    }

    private String populateAuthorities(Collection<? extends GrantedAuthority> authorities) {
        Set<String> auths = new HashSet<>();
        for (GrantedAuthority authority : authorities) {
            auths.add(authority.getAuthority());
        }
        return String.join( delimiter: ",", auths);
    }
}
```

AppConfig.java

```java
public class AppConfig {

    private final IUserRepo userRepo;

    @Bean
    public AuditorAware<Long> auditorProvider() {
        return () -> {
            Object principal = SecurityContextHolder.getContext().getAuthentication().getPrincipal();
            if (principal instanceof CustomUserDetails) {
                String username = ((CustomUserDetails) principal).getUsername(); // Email in your case
                Users user = userRepo.findByEmail(username);  // Assuming findByEmail exists in IUserRepo
                if (user != null) {
                    return Optional.of(Objects.requireNonNull(user.getId()));  // Return the user ID
                }
            }
            return Optional.empty();
        };
    }

    @Bean
    SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http.sessionManagement( SessionManagementConfigurer<HttpSecurity> management -> management.sessionCreationPolicy(SessionCreationPolicy.STATELESS));
        http.authorizeHttpRequests(
                        AuthorizationManagerRequestMat... authorize -> authorize
                                .requestMatchers("/api/admin**").hasAnyRole( ...roles: "RESTAURANT_OWNER", "ADMIN")
                                .requestMatchers("/api/**").authenticated()
                                .requestMatchers("/jwt/**").permitAll()
                                .anyRequest().permitAll())
                .addFilterBefore(new JwtTokenValidator(userRepo), BasicAuthenticationFilter.class)
                .csrf(AbstractHttpConfigurer::disable)
                .cors( CorsConfigurer<HttpSecurity> cors -> cors.configurationSource(corsConfigurationSource()));  // Enable CORS with the custom configuration
        return http.build();
    }
```

CustomerRestaurantController.java

```java
@Slf4j
@RestController
@RequestMapping("/api/restaurants")
@CrossOrigin(origins = "http://localhost:5173")
@RequiredArgsConstructor
public class CustomerRestaurantController {
    private final IRestaurantService restaurantService;
    private final IUserService userService;

    @GetMapping("/search")
    public ResponseEntity<List<Restaurant>> searchRestaurant(@RequestParam String keyword){
        List<Restaurant> restaurants = restaurantService.searchRestaurant(keyword);
        return new ResponseEntity<>(restaurants, HttpStatus.OK);
    }

    @GetMapping("/all")
    public ResponseEntity<List<Restaurant>> getAllRestaurant() {
        List<Restaurant> restaurants = restaurantService.getAllRestaurants();
        return new ResponseEntity<>(restaurants, HttpStatus.OK);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Restaurant> findRestaurantById(@PathVariable Long id) throws Exception {
        Restaurant restaurant = restaurantService.findRestaurantById(id);
        return new ResponseEntity<>(restaurant, HttpStatus.OK);
    }

    @PutMapping("/{id}/add-favorite")
    public ResponseEntity<RestaurantDTO> addToFavorite(@PathVariable Long id, @RequestHeader("Authorization") String jwt) throws Exception {
        Users user = userService.findUserByJwtToken(jwt);
        RestaurantDTO dto = restaurantService.addToFavorites(id, user);
        return new ResponseEntity<>(dto, HttpStatus.OK);
    }
```

Users.java

```java
package com.roshan.entity;

import ...

@EqualsAndHashCode(callSuper = true)
@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Users extends BaseEntity<Long> {

    private String fullName;

    @JsonProperty(access = Access.WRITE_ONLY)
    private String password;

    @Column(nullable = false, unique = true)
    private String email;

    @Enumerated(EnumType.STRING)
    @Column(nullable = false)
    private USER_ROLE role;

    @JsonIgnore
    @OneToMany(cascade = CascadeType.REMOVE, orphanRemoval = true)
    private List<Orders> orders = new ArrayList<>();

    @OneToMany(cascade = CascadeType.REMOVE, orphanRemoval = true)
    private List<Address> addresses = new ArrayList<>();
}
```

Project Alt+1 DetailsService.java

```java
import lombok.RequiredArgsConstructor;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;

@Service
@RequiredArgsConstructor
public class CustomUserDetailsService implements UserDetailsService {

    private final IUserRepo repo;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        Users user = repo.findByEmail(username);
        if (user != null) {
            USER_ROLE role = user.getRole();
            if (role == null) {
                role = USER_ROLE.ROLE_CUSTOMER;
            }
            List<GrantedAuthority> authorities = new ArrayList<>();
            authorities.add(new SimpleGrantedAuthority(role.toString()));
            return new CustomUserDetails(user.getId(), user.getEmail(), user.getPassword(), authorities);
        } else {
            throw new UsernameNotFoundException("User not found with email " + username);
        }
    }
```

IOrderItemRepo.java

```java
package com.roshan.repo;

import ...

public interface IOrderItemRepo extends JpaRepository<OrderItem, Long> {
    List<OrderItem> findByRestaurantOrderByIdAsc(Long restaurantId);

    List<OrderItem> findByCustomerId(Long userId);

    @Modifying
    @Transactional
    @Query("UPDATE OrderItem o SET o.orderStatus = :orderStatus WHERE o.id = :itemId")
    void updateStatus(Long itemId, String orderStatus);

}
```

BIBLIOGRAPHY

1. Kamal Acharya. School management system project report. Authorea. August 01, 2024. DOI: https://doi.org/10.22541/au.172254873.34023165/v1

2. Kamal Acharya. A CASE STUDY OF CINEMA MANAGEMENT SYSTEM PROJECT. Authorea. August 01, 2024. DOI: https://doi.org/10.22541/au.172254873.30191075/v1

3. Kamal Acharya. A CASE STUDY ON ONLINE TICKET BOOKING SYSTEM PROJECT. Authorea. August 01, 2024. DOI: https://doi.org/10.22541/au.172254872.26972790/v1

4. Kamal Acharya. Web chatting application project report management system. Authorea. August 01, 2024. DOI: https://doi.org/10.22541/au.172254871.18588592/v1

5. Kamal Acharya. RETAIL STORE MANAGEMENT SYSTEM PROJECT REPORT. Authorea. August 01, 2024. DOI: https://doi.org/10.22541/au.172254871.14590154/v1

6. Kamal Acharya. SUPERMARKET MANAGEMENT SYSTEM PROJECT REPORT. Authorea. August 01, 2024. DOI: https://doi.org/10.22541/au.172252491.19145062/v1

7. Kamal Acharya. SOCIAL MEDIA MANAGEMENT SYSTEM PROJECT REPORT. Authorea. August 01, 2024. DOI: https://doi.org/10.22541/au.172252491.11210579/v1

8. Kamal Acharya. Online music portal management system project report. Authorea. August 01, 2024. DOI: https://doi.org/10.22541/au.172252488.89734698/v1

9. Kamal Acharya. COLLEGE BUS MANAGEMENT SYSTEM PROJECT REPORT. Authorea. July 31, 2024. DOI: https://doi.org/10.22541/au.172245277.70798942/v1

10. Kamal Acharya. AUTOMOBILE MANAGEMENT SYSTEM PROJECT REPORT. Authorea. July 31, 2024. DOI: https://doi.org/10.22541/au.172245276.67982593/v1

11. Kamal Acharya. Ludo management system project report. Authorea. July 31, 2024. DOI: https://doi.org/10.22541/au.172243999.98091616/v1

12. Kamal Acharya. Literature online quiz system project report. Authorea. July 31, 2024 DOI: https://doi.org/10.22541/au.172243825.53562953/v1

13. Kamal Acharya. CHAT APPLICATION THROUGH CLIENT SERVER MANAGEMENT SYSTEM PROJECT. Authorea. July 29, 2024. DOI: https://doi.org/10.22541/au.172228527.74316529/v1 2025.

14. Acharya, Kamal, Online Job Portal Management System (May 5, 2024). Available at SSRN: https://ssrn.com/abstract=4817534 or http://dx.doi.org/10.2139/ssrn.4 817534

15. Acharya, Kamal, Employee leave management system. (May 7, 2024). Available at SSRN: https://ssrn.com/abstract=4819626 or http://dx.doi.org/10.2139/ssrn.4 819626

16. Acharya, Kamal, Online electricity billing project report. (May 7, 2024). Available at SSRN: https://ssrn.com/abstract=4819630 or http://dx.doi.org/10.2139/ssrn.4 819630

17. Acharya, Kamal, POLICY MANAGEMENT SYSTEM PROJECT REPORT. (December 10, 2023). Available at SSRN: https://ssrn.com/abstract=4831694 or http://dx.doi.org/10.2139/ssrn.4 831694